Funcionalidades de la aplicación

- El usuario se puede registrar si no tiene cuenta a través de un formulario.
- Cualquier usuario una vez iniciada sesión se le enseña la página principal con sus inversiones realizadas. Si hace click en el enlace del ticker de la compañia, ésta le lleva a una página de detalle con su descripción.
- Asimismo, el usuario, puede ver una lista de compañías disponibles.

Funcionalidades adicionales:

- Se valorará incluir compañías reales y obtener su valoración a través de una API pública. de este modo, cada usuario puede ver la valoración de sus inversiones en tiempo real.
- Esto presenta un problema de implementación de nuevas compañías. Se valorará si sólo incluír inversiones nuevas, o también compañías.
- Para su diseño se utilizará un framework de CSS, en este caso Bootstrap o Tailwind.

Diseño de la base de Datos:

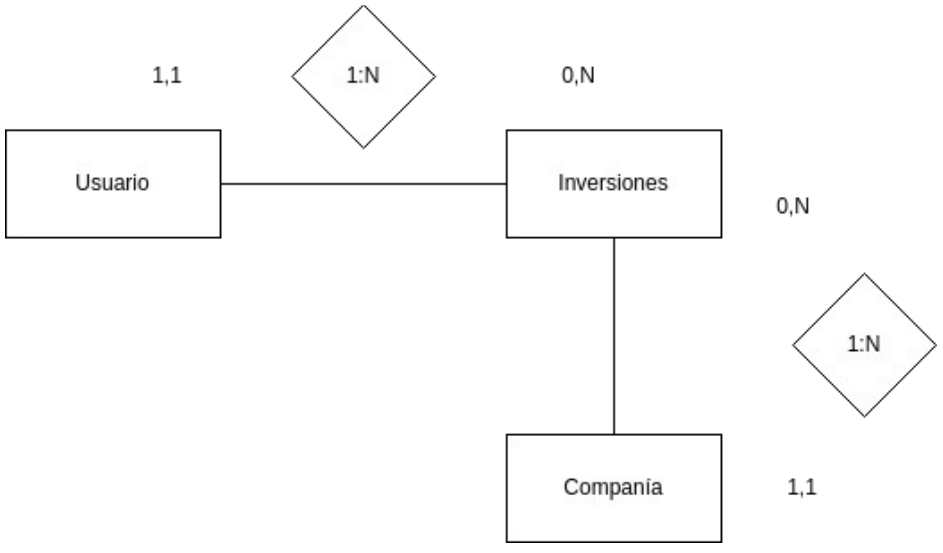
La relación entre **Usuario** e **Inversión** es de **uno a muchos (1:N)**, ya que:

- Un **Usuario** puede tener **cero o más** inversiones (0..N).
- Cada **Inversión** pertenece **a un solo Usuario** (1..1).
- **Cada inversión** pertenece a **una única compañía** (**1,1**).
- **Cada compañía** puede tener **cero o muchas inversiones** (**0,N**).

Claves Foráneas:

- La tabla de inversiones necesita una foreign key tanto como de usuarios como de compañías, para poder normalizarse.
- Se usa `ON DELETE CASCADE` en ambas claves foráneas para mantener la integridad referencial.

Esto quiere decir que si se elimina un usuario, se eliminarán sus inversiones. De tal modo que si se elimina una compañía, se eliminarán también sus inversiones asociadas.



ESQUEMA RELACIONAL

```python
class Usuario(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=100)
    email = models.EmailField(max_length=100, unique=True)
    password = models.CharField(max_length=100)

    def __str__(self):
        return self.name


class Corporation(models.Model):

    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=100)
    ticker = models.CharField(max_length=10, unique=True)
    description = models.TextField()
    country = models.CharField(max_length=100)
    sector = models.CharField(max_length=100, blank=True)

    def __str__(self):
        return self.name


class Investment(models.Model):
    id = models.AutoField(primary_key=True)
    user = models.ForeignKey(
        "stocks.Usuario",
        on_delete=models.CASCADE,
        default=1,
        related_name="investments",
    )
    corporation = models.ForeignKey(
        "stocks.Corporation",
        on_delete=models.CASCADE,
        default=1,
        related_name="investments",
    )
    stocks_amount = models.DecimalField(max_digits=10,
decimal_places=2)
    price_at_purchase = models.DecimalField(max_digits=10,
 decimal_places=2)

    @property
    def total_investment(self):
        return self.stocks_amount * self.price_at_purchase

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"{self.user.name} - {self.corporation.name
} ({self.total_investment}€)"
```